Explain all the following function in brief

1. **\_\_init\_\_(self) (ie.constructor).**

* The \_\_init\_\_ method is the Python equivalent of the [C++ constructor](https://www.udacity.com/blog/2021/03/what-is-a-constructor-in-c.html?utm_source=rss&utm_medium=rss&utm_campaign=what-is-a-constructor-in-c) in an object-oriented approach. The \_\_init\_\_ function is called every time an object is created from a class. The \_\_init\_\_ method lets the class initialize the object’s attributes and serves no other purpose. It is only used within classes.

**class Dog:**

**def \_\_init\_\_(self,dogBreed,dogEyeColor):**

**self.breed = dogBreed**

**or = dogEyeColor**

* Use the \_\_init\_\_() method to initialize the object’s attributes.
* The \_\_init\_\_() doesn’t create an object but is automatically called after the object is created.

1. **\_\_str\_\_(self) method**:

* The **\_\_str\_\_ method** in Python represents the class objects as a string – it can be used for classes. The \_\_str\_\_ method should be defined in a way that is easy to read and outputs all the members of the class. This method is also used as a debugging tool when the members of a class need to be checked.
* The \_\_str\_\_ method is called when the following functions are invoked on the object and return a string:
* print()
* str()

class Person:

1. class Person:
2. ...: pass
3. ...:
5. In [2]: p = Person()
7. In [3]: print p
8. <\_\_main\_\_.Person instance at 0x7faffb3ac5f0>
10. In [4]: class Person:
11. ...: def \_\_str\_\_(self):
12. ...: return 'Person class'
13. In [5]: p = Person()
15. In [6]: print p
16. Person class
17. **\_\_del\_\_(self) (ie.destructor):**

* The users call Destructor for destroying the object. In Python, developers might not need destructors as much it is needed in the C++ language. This is because Python has a garbage collector whose function is handling memory management automatically.
* The **\_\_del\_\_()** function is used as the destructor function in [Python](https://www.javatpoint.com/python-tutorial). The user can call the **\_\_del\_\_()** function when all the references of the object have been deleted, and it becomes garbage collected.

class Animals:

   #  we will initialize the class

    def \_\_init\_\_(self):

        print('The class called Animals is CREATED.')

    # now, we will Call the destructor

    def \_\_del\_\_(self):

        print('The destructor is called for deleting the Animals.')

object = Animals()

del object

**Output:**

The class called Animals is CREATED.

The destructor is called for deleting the Animals.